Kinds of Plans

Start State

On(C, A)
On(A, Table)
On(B, Table)
Clear(C)
Clear(B)
Block(A)
...

Sequential Plan

MoveToTable(C,A) > Move(B,Table,C) > Move(A,Table,B)

Partial-Order Plan

MoveToTable(C,A) > Move(A,Table,B)
Move(B,Table,C)
Forward Search

Start State

On(C, A)
On(A, Table)
On(B, Table)
Clear(C)
Clear(B)
Block(A)
...

MoveToBlock(C,A,B)

MoveToBlock(B,Table,C)

MoveToTable(C,A)

On(C, A)
On(A, Table)
On(B, Table)
Clear(C)
Clear(B)
Block(A)
...
+Clear(A)
+On(C, Table)

Applicable actions
Backward Search

ACTION: MoveToBlock(b,x,y)
PRECONDITIONS: On(b,x), Clear(b), Clear(y),
Block(b), Block(y), (b\(\neq\)x), (b\(\neq\)y), (x\(\neq\)y)
POSTCONDITIONS: On(b,y), Clear(x)
\(\neg\text{On}(b,x), \neg\text{Clear}(y)\)

MoveToBlock(A, Table, B)
MoveToBlock(A, x', B)

On(B, C)
On(A, B)
+On(A, Table)
+Clear(A)
+Clear(B)
+…

Goal State
On(B, C)
On(A, B)

Relevant actions

\[ g' = (g - \text{ADD}(a)) \cup \text{Precond}(a) \]
Heuristics: Ignore Preconditions

- Relax problem by ignoring preconditions
  - Can drop all or just some preconditions
  - Can solve in closed form or with set-cover methods

**Action (Slide(t, s₁, s₂),**

**PRECOND:** \(\text{On}(t, s₁) \land \text{Tile}(t) \land \text{Blank}(s₂) \land \text{Adjacent}(s₁, s₂)\)

**EFFECT:** \(\text{On}(t, s₂) \land \text{Blank}(s₁) \land \neg\text{On}(t, s₁) \land \neg\text{Blank}(s₂)\)
Heuristics: No-Delete

- Relax problem by not deleting falsified literals
  - Can’t undo progress, so solve with hill-climbing (non-admissible)

ACTION: MoveToBlock(b,x,y)

- PRECONDITIONS: On(b,x), Clear(b), Clear(y), Block(b), Block(y), (b ≠ x), (b ≠ y), (x ≠ y)
- POSTCONDITIONS: On(b,y), Clear(x), On(b,x), ¬Clear(y)
Heuristics: Independent Goals

- **Independent subgoals?**
  - Partition goal literals
  - Find plans for each subset

Goal State

- On(B, C)
- On(A, B)

On(A, B) → On(B, C)
Planning “Tree”

Start: HaveCake

Goal: AteCake, HaveCake

Action: Eat
Pre: HaveCake
Add: AteCake
Del: HaveCake

Action: Bake
Pre: ¬HaveCake
Add: HaveCake
Reachable State Sets

Have=T, Ate=F

{Eat} {} 

Have=F, Ate=T

{Bake} {} 

Have=T, Ate=F

Have=F, Ate=T

Have=F, Ate=T

Have=T, Ate=T

Have=F, Ate=F

Have=T, Ate=F

Have=F, Ate=T

Have=F, Ate=T

Have=T, Ate=F

{Eat} {}
Approximate Reachable Sets

- Have=T, Ate=F
- Have={T}, Ate={F}

- Have=F, Ate=T  Have=T, Ate=F
- Have={T,F}, Ate={T,F}

- Have=F, Ate=T  Have=F, Ate=T  Have=T, Ate=F
- Have={T,F}, Ate={T,F}

(Have, Ate) not (T,T)  (Have, Ate) not (F,F)
Planning Graphs

Start: HaveCake

Goal: AteCake, HaveCake

Action: Eat
  Pre: HaveCake
  Add: AteCake
  Del: HaveCake

Action: Bake
  Pre: \neg HaveCake
  Add: HaveCake

S_0 \quad A_0 \quad S_1
Mutual Exclusion (Mutex)

**NEGATION**

Literals and their negations can’t be true at the same time

\[ \neg P \]

\[ P \]

\[ \neg \neg P \]

\[ \neg \neg \neg P \]

\[ \neg \neg \neg \neg P \]
Mutual Exclusion (Mutex)

INCONSISTENT EFFECTS
An effect of one negates the effect of the other

\[ S_0 \quad A_0 \quad S_1 \]

- HaveCake
- Eat
- \( \neg \text{AteCake} \)
- \( \neg \text{HaveCake} \)
- AteCake
- \( \neg \text{AteCake} \)
Mutual Exclusion (Mutex)

INCONSISTENT SUPPORT
All pairs of actions that achieve two literals are mutex
Mutual Exclusion (Mutex)

COMPETITION
Preconditions are mutex; cannot both hold

INCONSISTENT EFFECTS
An effect of one negates the effect of the other
Mutual Exclusion (Mutex)

INTERFERENCE
One deletes a precondition of the other

S_1

\begin{align*}
\neg \text{HaveCake} \\
\neg \text{AteCake}
\end{align*}


S_2

\begin{align*}
\text{HaveCake} \\
\text{AteCake}
\end{align*}

\begin{align*}
\text{AteCake} \\
\neg \text{AteCake}
\end{align*}

\begin{align*}
\neg \text{HaveCake} \\
\neg \text{HaveCake}
\end{align*}

\begin{align*}
\text{Sell} \\
\text{Bake}
\end{align*}

\begin{align*}
\text{Eat}
\end{align*}

\begin{align*}
\text{S_1} \\
\text{A_1} \\
\text{S_2}
\end{align*}
Observation 1

Propositions monotonically increase
(always carried forward by no-ops)
Observation 2

Actions monotonically increase
(if they applied before, they still do)
Observation 3

Proposition mutex relationships monotonically decrease
Observation 4

Action mutex relationships monotonically decrease
Observation 5

• Claim: planning graph “levels off”
  – After some time $k$ all levels are identical
  – Because it’s a finite space, the set of literals cannot increase indefinitely, nor can the mutexes decrease indefinitely

• Claim: if goal literal never appears, or goal literals never become non-mutex, no plan exists
  – If a plan existed, it would eventually achieve all goal literals (and remove goal mutexes – less obvious)
  – Converse not true: goal literals all appearing non-mutex does not imply a plan exists
Heuristics: Level Costs

• Planning graphs enable powerful heuristics
  – Level cost of a literal is the smallest S in which it appears
  – Max-level: goal cannot be realized before largest goal conjunct level cost (admissible)
  – Sum-level: if subgoals are independent, goal cannot be realized faster than the sum of goal conjunct level costs (not admissible)
  – Set-level: goal cannot be realized before all conjuncts are non-mutex (admissible)
Graphplan

- Graphplan directly extracts plans from a planning graph
- Graphplan searches for **layered plans** (often called parallel plans)
  - More general than totally-ordered plans, less general than partially-ordered plans
- A layered plan is a sequence of **sets** of actions
  - actions in the same set must be compatible
  - all sequential orderings of compatible actions gives same result

```
move(A,B,TABLE)
moves(C,D,TABLE)
moves(B,TABLE,A)
moves(D,TABLE,C)
```

**Layered Plan:** (a two layer plan)

\[
\begin{align*}
\text{Layered Plan:} & \quad (\text{a two layer plan}) \\
& \quad \begin{cases}
\{ \text{move}(A, B, \text{TABLE}) \} \\
\{ \text{move}(C, D, \text{TABLE}) \}
\end{cases} \quad , \quad \begin{cases}
\{ \text{move}(B, \text{TABLE}, A) \} \\
\{ \text{move}(D, \text{TABLE}, C) \}
\end{cases}
\end{align*}
\]
Solution Extraction: Backward Search

Search problem:
Start state: goal set at last level
Actions: conflict-free ways of achieving the current goal set
Terminal test: at $S_0$ with goal set entailed by initial planning state

Note: may need to start much deeper than the leveling-off point!

Caching, good ordering is important